Bringing Interactivity to Research Papers and Presentations with Chameleon

Damien Masson, Sylvain Malacria, Edward Lank, Géry Casiez

Bringing Interactivity to Research Papers and Presentations with Chameleon

Damien Masson
University of Waterloo, Canada
Inria, France
dmasson@uwaterloo.ca

Edward Lank
University of Waterloo, Canada
Inria, France
lank@uwaterloo.ca

Sylvain Malacria
Inria, France
Univ. Lille, UMR 9189 - CRIStAL
sylvain.malacria@inria.fr

Géry Casiez
Univ. Lille, UMR 9189 - CRIStAL
Inria, France
University of Waterloo, Canada
Institut Universitaire de France
gery.casiez@univ-lille.fr

To cite this work, please refer to the full publication [5] in Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems (CHI 2020).
Authors' version.
CHI '20 Extended Abstracts, April 25–30, 2020, Honolulu, HI, USA.

http://dx.doi.org/10.1145/3334480.3383173

Abstract
Chameleon is a software that combines computer vision feature-matching algorithms with an open database format to allow the incorporation of dynamic HTML5 interactive content over any type of document (e.g. PDF files, PowerPoint documents, etc.) without modifying existing applications or the source document. It thus allows the provision and viewing of an enhanced version of a research paper with embedded interactive demonstrations or videos. It can also be used to perform live demonstrations of interaction techniques while giving a presentation without having to switch tools.

Author Keywords
Augmented documents; feature matching; interactivity

CCS Concepts
•Human-centered computing → **Interactive systems and tools; •Applied computing** → *Multi / mixed media creation;*

Introduction
Documents such as presentations, instruction manuals, and research papers are disseminated using various file formats, many of which barely support the incorporation of interactive content. However, even a task as simple as describing an interaction technique or interactive scenario

Example usage: authors
Andy is an HCI researcher who has published a research *paper* introducing a new interaction technique. He also has a *project page* on his personal website, including a *demo*, an *interactive chart* and a *video* of the interaction technique. He wants to add these interactive figures within the research paper.

For that, he starts Chameleon's REGISTRATION TOOL, draws a bounding box over the *figures* illustrating respectively the interaction technique, the chart and the static image and links them to their online interactive counterparts.

requires careful wording when presented in static form, often requiring a *comic strip* figure describing the interaction in several steps, rather than letting the reader *experience* it. Similarly, reporting scientific results often requires an author to make decisions toward which data, and in which form, to report in a paper or while giving a research talk, rather than interacting with the data or allowing a reader to interact with alternative visualizations or analyses of the data.

To address this lack of interactivity, we present Chameleon (Figure 1), a tool that combines feature-matching with an open database format to allow the incorporation and overlaying of dynamic content over any type of document (e.g. PDF files, Microsoft Word and PowerPoint documents, eBooks, etc.) without modifying existing applications or the source document. Chameleon can be used to upgrade these documents with user-generated HTML5 interactive content, thus allowing the simplified provision and viewing of an enhanced version of a research paper with embedded interactive demonstrations and videos. It can also be used to perform live demonstrations of interaction techniques while giving a presentation without having to switch tools (see example usage on the left).

Chameleon

The core idea of Chameleon is to augment pre-registered figures of digital documents by layering HTML5 resources over the figures (Figure 1). It uses the SURF algorithm to identify the pre-registered figures, and overlay on them an undecorated WebView that runs the HTML5 resource. Chameleon has been developed in C++ using the Qt 5.12 framework and OpenCV 4.0.1, together with platform dependent APIs. OpenCV provides the SURF implementation while Qt provides the WebKit engine to render and interact with the HTML resources overlaid on augmented figures. Chameleon currently runs on macOS and Linux using

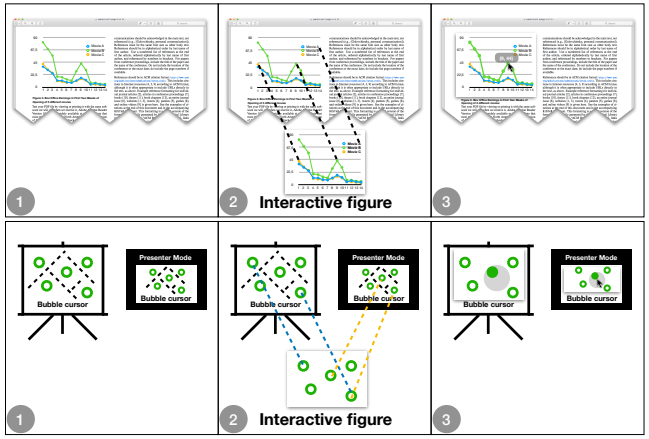

Figure 1: Static documents such as PDF (top) and presentations (bottom) can be augmented with Chameleon by locating corresponding interactive figures in the Chameleon database and overlaying them on top of the existing documents.

respectively Cocoa and X11 APIs[1] and could also be implemented for Windows by simply replacing the platform dependent API calls.

More precisely, Chameleon comprises of two main parts representing its two primary use-cases:

- REGISTRATION TOOL: used to register a figure from a document with its augmented version
- AUGMENTATION TOOL: used to analyse opened documents and *augment* them if *augment-able* figures are detected.

[1] Source code available at ns.inria.fr/loki/chameleon

Example usage: readers
Carl installed Chameleon on his laptop. He is interested by the interaction technique proposed by Andy and decides to learn more about it by downloading and reading the research paper. Carl then opens the paper using his favourite PDF viewer and sees that some figures are appearing through a fade in. While skimming the paper, Carl notices that those new figures are interactive. He immediately tries the *demo* of the interaction technique to grasp the idea.

Carl is then interested by comparing the performance of this technique with the performance of a technique he is developing. Using the interactive charts showing the results, Carl is able to read the exact numbers needed for his comparison. Finally, to make sure that he did not miss anything, Carl watches the video within the paper.

The registration tool
The registration tool provides the interface to select a portion of a static document that will be registered in the database. To accomplish this, the user first opens a document that contains the figure, and navigates in the document to where the figure is located. Using the *Register* command in the Chameleon menu or the associated keyboard shortcut (CTRL+F1), she opens a screenshot of the current viewport in a new window in which she can select the region she wants to augment by dragging the mouse, highlighting the corresponding zone. The interface also provides a text field to enter the URL of the augmented resource. After validation, the augmentation is stored in a database. In order to compress the size of the database, we do not store the image of the figure but, instead, only store the pre-computed regions and descriptors of the image as provided by feature matching algorithms along with the document file size, its MD5, the dimensions of the selected area, and the url of the augmented resource. One potential downside of this approach is that, if an augmented document is modified, its MD5 changes. To address this issue, the registration tool also tracks modifications to augmented documents using DTrace [7], and, when an augmented document is modified, the user is prompted to either ignore or register the new, edited version of the document.

The augmentation tool
The augmentation tool runs in background and loops on the following six steps (summarized on the left):

STEP 1: The augmentation tool monitors file access on the system using DTrace [7] on Linux and macOS. It retrieves the size of each opened file and queries the database to get the list of files with augmented figures of the same file sizes. Finally it computes the MD5 for the remaining files to filter the remaining documents.

STEP 2: The augmentation tool takes a screenshot of every window containing documents to be augmented, using the XGETIMAGE [3] function on Linux, and CGWINDOWLISTCREATEIMAGE [2] on macOS. These functions provide the content of a window even if partially hidden, without decorations.

STEP 3: Chameleon uses feature matching algorithms to determine regions and descriptors in the screenshots and detect potential figures to be augmented. This step takes the longest time in the augmentation process. We therefore re-compute regions and descriptors only when the window is scrolled or resized. The feature matching algorithm then tries to associate the regions found with the ones stored in the database to detect the figures present and determine their location and size.

STEP 4: For each figure found, the augmentation tool opens an undecorated window with transparent background. These windows cannot be focused, resized or moved. Their size and position are dynamically updated to be the same as the figures to augment. Each window contains a QWEBENGINEVIEW [6] used to load the remote HTML content located at the URL associated with the figure to augment. Users can interact with augmented figures using the mouse cursor. In addition, they can hide and show augmented figures by using a button located at the top left corner useful to review the original static figure. They can also get additional information about the augmented figure in a drop down menu and switch between different augmented figures available in the database (in the case where multiple augmented figures exist for an individual figure in the source document). Augmented figures are pre-loaded in background when opened files containing augmented figures are found in step 1. The window is first fully invisible and made visible when a figure is found in step 3. We use a 500ms delay and a 1000ms

Example usage: presenters
Andy is invited to give a talk about his research paper. While preparing his slides, he would like to highlight some specific values in the results showed by the chart within the paper. Instead of creating multiple slides, Andy decides to leverage Chameleon to include his already existing interactive chart in his presentation. Finally, Andy decides to include a live demo of the interactive technique at the beginning of the presentation. Again, he uses Chameleon as a way to perform the demonstration without leaving the presentation mode.

fade-in effect, tuned via informal pilot testing, to inform the user that a figure is augmented and let her verify that the augmented figure matches the static one..

STEP 5: Linux provides an API called AT-SPI (Assistive Technology Service Provider Interface) [4] that can be used to notify when a window is scrolled or resized. macOS provides the same services through its Accessibility API [1]. The translation of windows is tracked by pooling their position when a mouse drag event is detected. Each window displaying an augmented figure registers callbacks to be notified when one of the properties of the window displaying the document changes. These callbacks are called with a small latency making it possible to re-position and resize the augmented figures with artefacts that are hard to notice. When a figure is cropped (typically during or after scrolling), we apply a mask on the augmented window so that it is cropped in the same way and does not extend outside the document window. This step is not mandatory for using Chameleon, as overlaying interactive figures over a static document can be achieved without it, but it results in a smoother integration of the augmented figures.

STEP 6: When Chameleon detects multiple identical figures on the screen, the overlaid augmented versions can be synchronized. The augmented figure with the highest resolution is tagged as the main figure; the other ones are considered copies and resized to match their overlaid static figure resolutions. All the events received by the copies, e.g. keyboard and mouse events, are redirected onto the main figure. This step, enabled by default, can be disabled via Chameleon's menu.

Conclusion

We presented Chameleon, a system that leverages computer-vision based feature matching and a database of active content to graft interactivity onto static figures within a corpus of documents. Our implementation works with macOS and Linux desktop applications, and only requires to replace the platform dependent API calls in order to adapt it for Microsoft Windows. Chameleon is distributed as an open source software.

REFERENCES

[1] Apple. 2007a. Accessibility on macOS. (2007). Retrieved April 7th, 2019 from https://developer.apple.com/accessibility/macos/

[2] Apple. 2007b. CGWindowListCreateImage. (2007). Retrieved April 7th, 2019 from https://developer.apple.com/documentation/coregraphics/1454852-cgwindowlistcreateimage

[3] die.net. 1987. XGetImage. (1987). Retrieved April 7th, 2019 from https://linux.die.net/man/3/xgetimage

[4] GNOME. 2011. Assistive Technology Service Provider Interface. (2011). Retrieved April 7th, 2019 from https://github.com/GNOME/at-spi2-atk

[5] Damien Masson, Sylvain Malacria, Edward Lank, and Géry Casiez. 2020. Chameleon: Bringing Interactivity to Static Digital Documents. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*. Association for Computing Machinery, New York, NY, USA, Article Paper 432, 13 pages. DOI: http://dx.doi.org/10.1145/3313831.3376559

[6] Qt. 2008. QWebEngineView. (2008). Retrieved April 7th, 2019 from https://doc.qt.io/qt-5/qwebengineview.html

[7] Wikipedia. 2005. DTrace. (2005). Retrieved April 7th, 2019 from https://en.wikipedia.org/wiki/DTrace